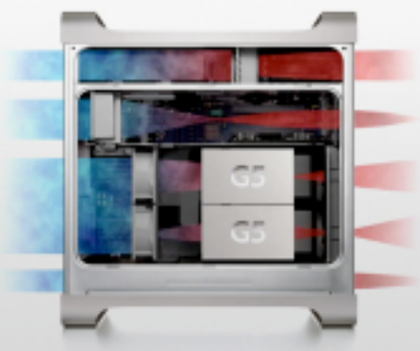
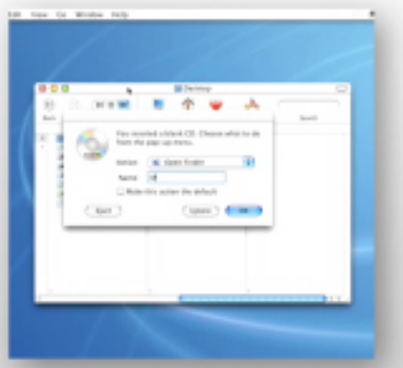
FireWire In-Depth

Session 505



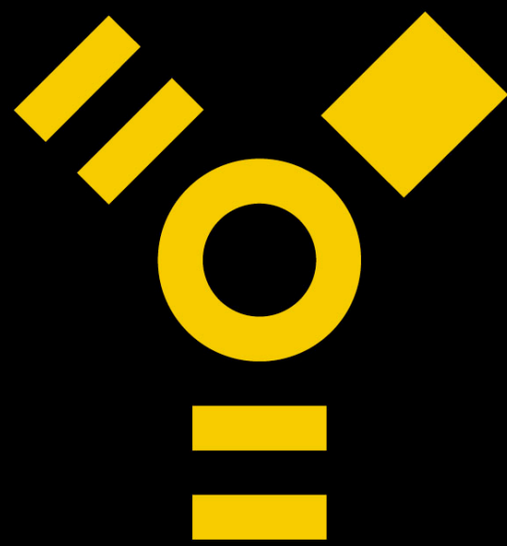


FireWire In-Depth

Eric Anderson
Manager—FireWire Software
ewa@apple.com

Introduction

- This session is about FireWire software
  - What layers Apple provides
  - How to use Apple's APIs
  - What is new and improved since 2002

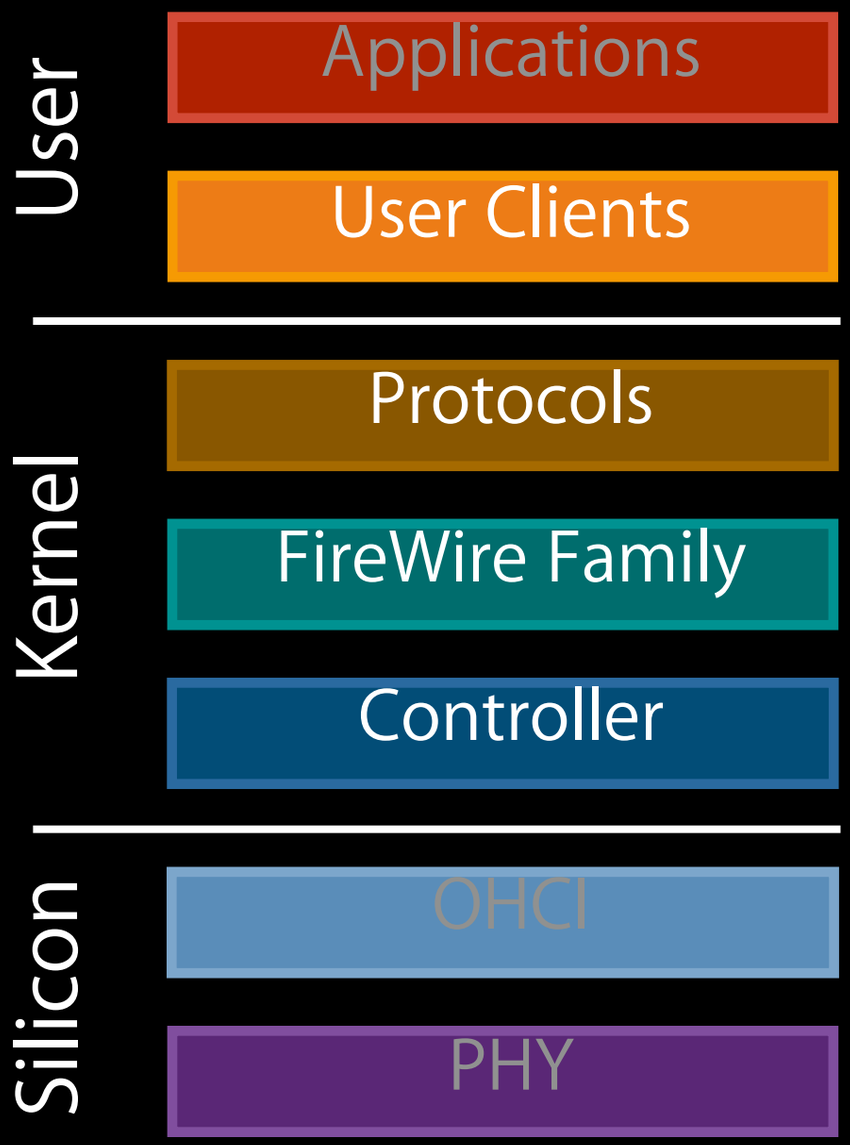


What You Will Learn

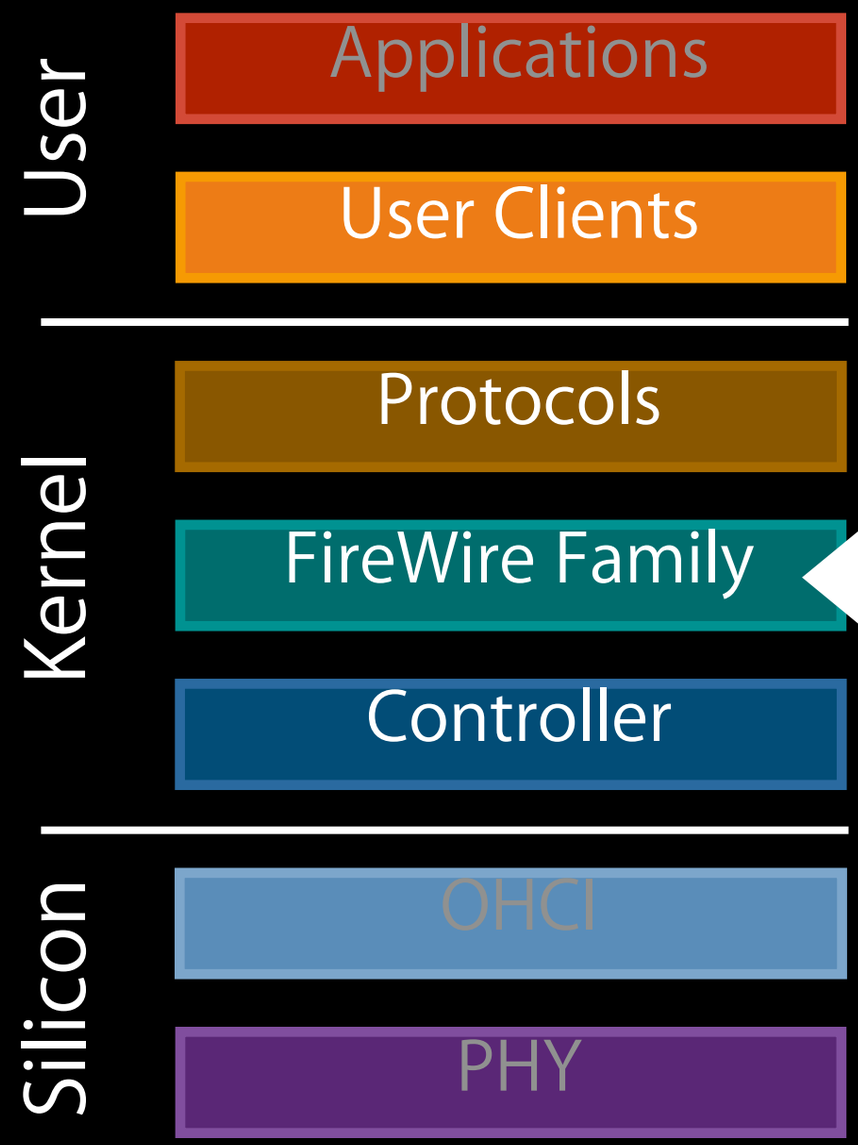
- FireWire software architecture
- Where to develop software
- TCP/IP on FireWire
- Isochronous transfer
- AV/C services
- SBP-3 support
- Tools and support

FireWire Software Architecture

Mac OS X FireWire

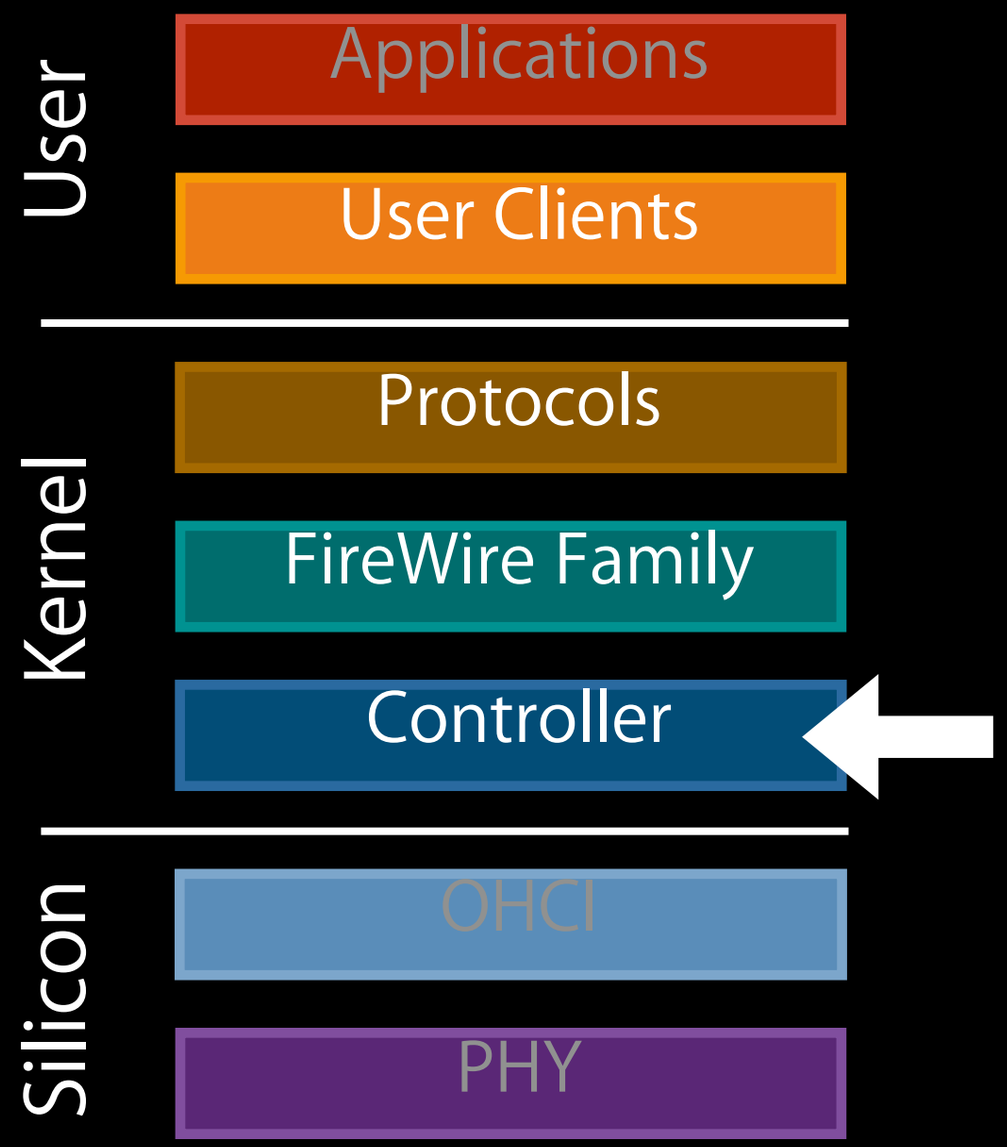


- Service layers in kernel
  - OHCI
  - Family
  - Protocols (SBP, AVC, IP)
- User clients
  - Application access
  - User-space drivers/plug-ins



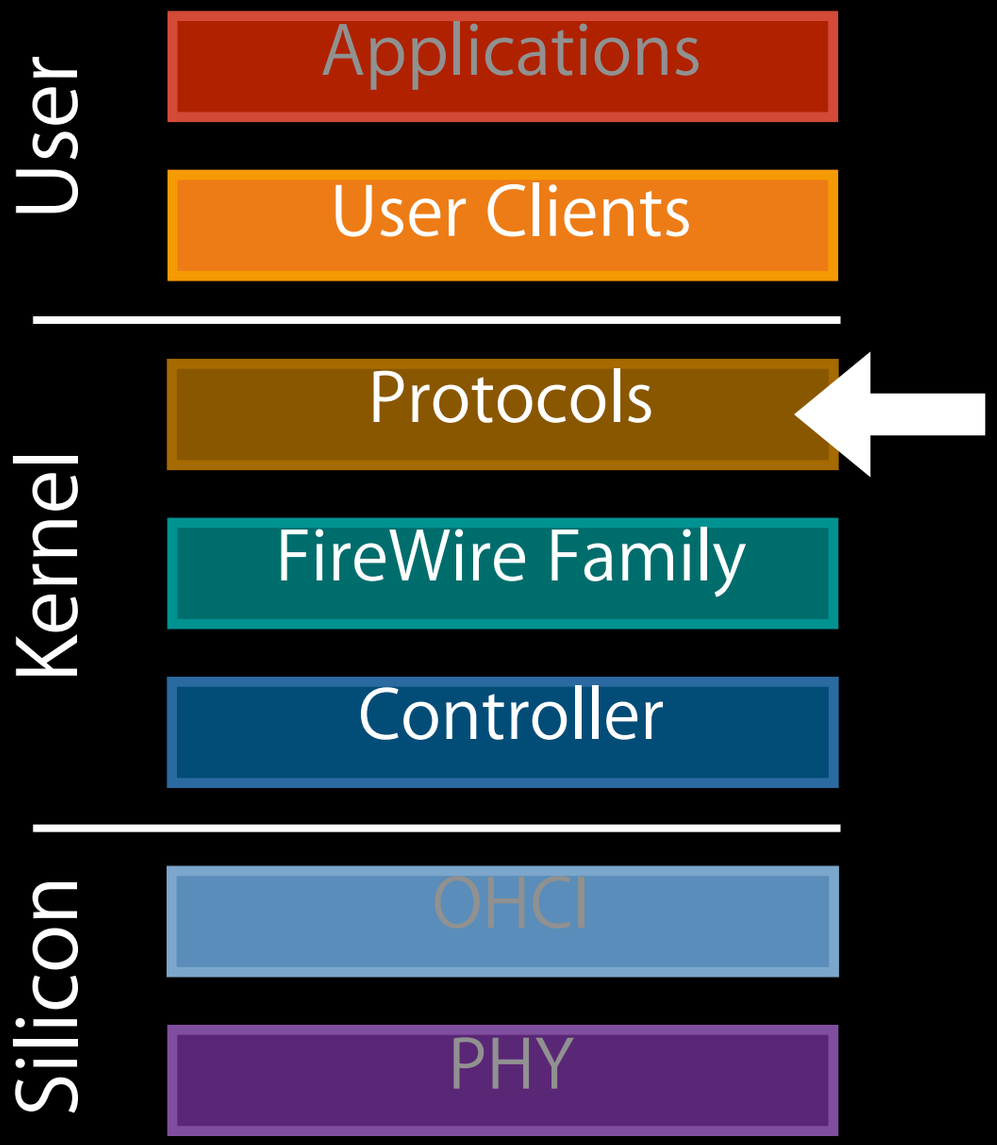
FireWire Family

- Multiplexing layer for all FireWire resources
  - Devices
  - Drivers
  - Protocols
  - Applications
- IOFireWireFamily.kext allows everyone to share access to one bus
  - Or more than one bus



Controller

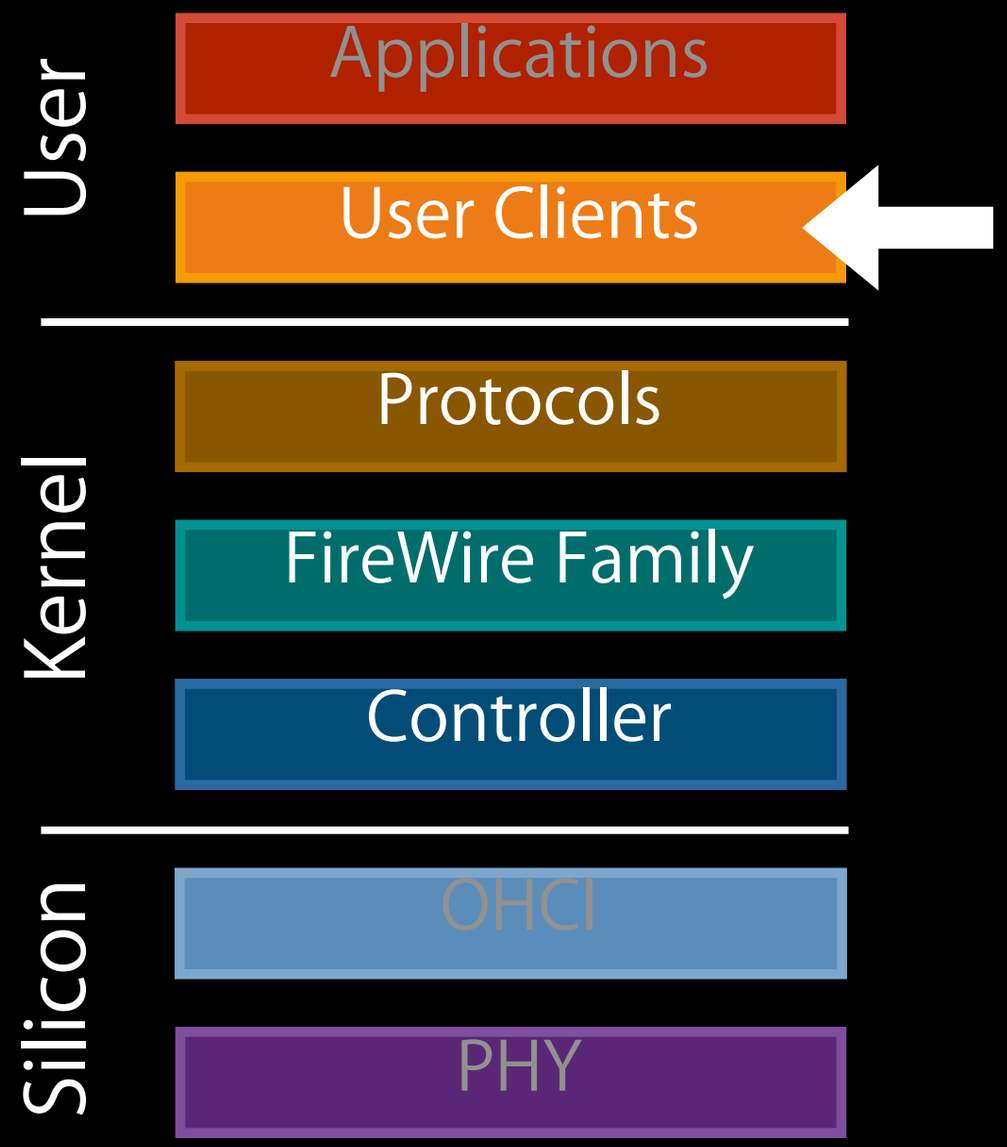
- Device driver for FireWire hardware interface
  - AppleFWOHCI.kext
  - AppleLynx.kext (older)
- Integrated, PCI, or CardBus
  - No quantity limit
- Cannot be subclassed
  - OHCI and PHY must comply with standards



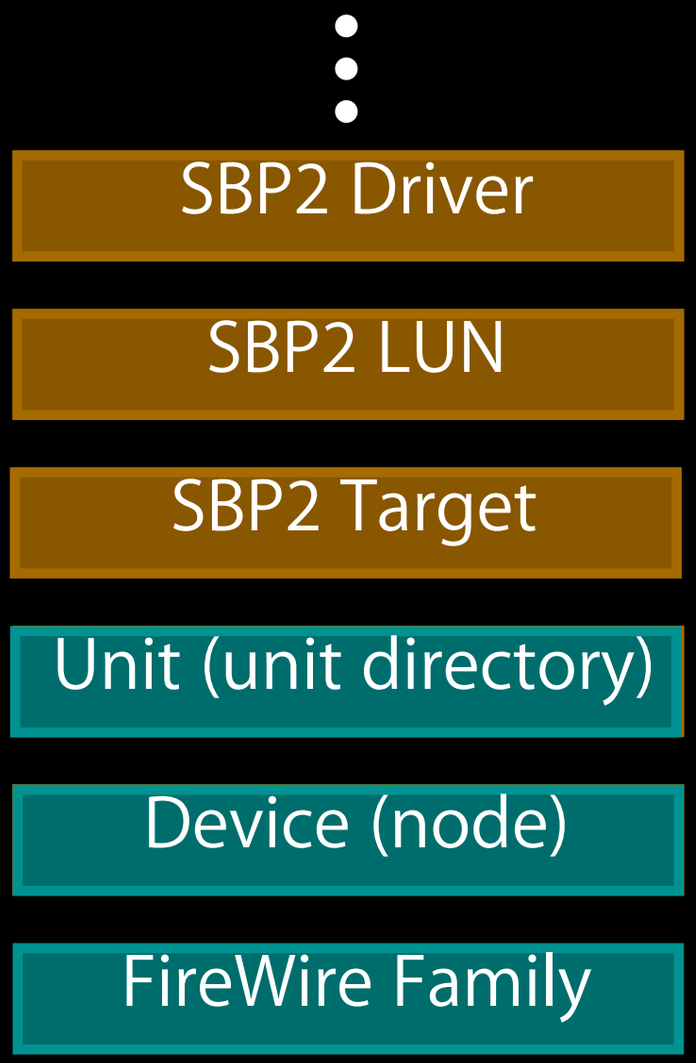
Protocols

- Specialized service layers for different device types
  - Serial Bus Protocol
  - Audio/Video Command
  - Internet Protocol (TCP/IP)
- Protocol-specific matching
  - SBP by LUN device type
  - AVC by Unit or SubUnit

User Clients

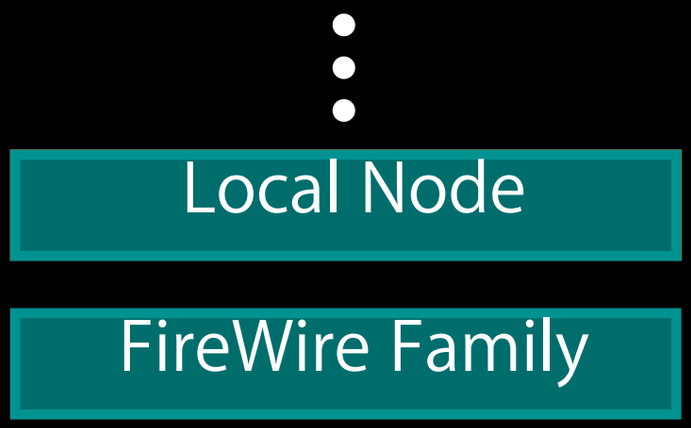


- User space (Application) access to kernel services
  - Nearly all FireWire services are available in user space
  - Direct (asynch/isoch) and protocol-specific APIs
- Writing and debugging FireWire software in user space is much easier than doing it in the kernel



Driver Matching

- Drivers are matched by standard IO Kit services
- FireWire Family publishes general data in IO Registry
  - Vendor ID, Model, etc.
- Protocol layers publish additional information
  - SBP: Command set, type
  - AVC: Unit, SubUnits
- Matching is iterative



The Local Node

- Every FireWire bus (IOFireWireFamily) has a Local Node in the Registry
  - Represents the Mac itself
  - Always present
- Match on the Local Node for protocols and some multi-node applications
  - User client access is available

Where to Develop Software

User Space (First Choice)

- Nearly all FireWire services are available through one or more User Clients
  - IOFireWireLib for all-purpose direct FireWire access
  - IOFireWireAVCLib
  - IOFireWireSBP2Lib
    - SCSITaskUserClient for many SBP-2 devices
- Lots of sample code in the FireWire SDK

  /Developer/Examples/IOKit/firewire/

Kernel (Last Choice)

- All FireWire services available
- Subclassing may be possible
  - Apple uses subclasses a lot for FireWire mass storage
- Can support other kernel services
  - Boot device, networking, possibly audio
- Limited sample code
  - FireWire family and protocol layers are in Darwin
- Debugging and testing is difficult

User Space Sample Code

- Install SDK 16 to get /Developer/Examples/IOKit/firewire/

AVCBrowser
DVHSCap
FWLib
FWNotifier
FWPtP
FWSampleProject
FWUtil
FireWireMPEG
IOFireWireLibBasics
IOFireWireLibIsochTest
IOFireWireLibPacketQueueTest
IOFireWireLibVersionTest
IsocTest
ResetStorm
SBP2SampleProject
Transcalent

Storage and SCSI-Type Devices

- SCSITaskUserClient for many SBP-2 devices

  /System/Library/Extensions/
  IOSCSIArchitectureModelFamily.kext/Contents/PlugIns

  – Printers, scanners, still cameras, etc.
- Disc Recording APIs for CD-R, DVD-R, etc.

  /Developer/Documentation/Carbon/multimedia/
  discrecording/discrecordingserv.html
- Use FireWire APIs or possibly SAM subclass if your device has SBP-2 bugs or limitations

New TCP/IP on FireWire

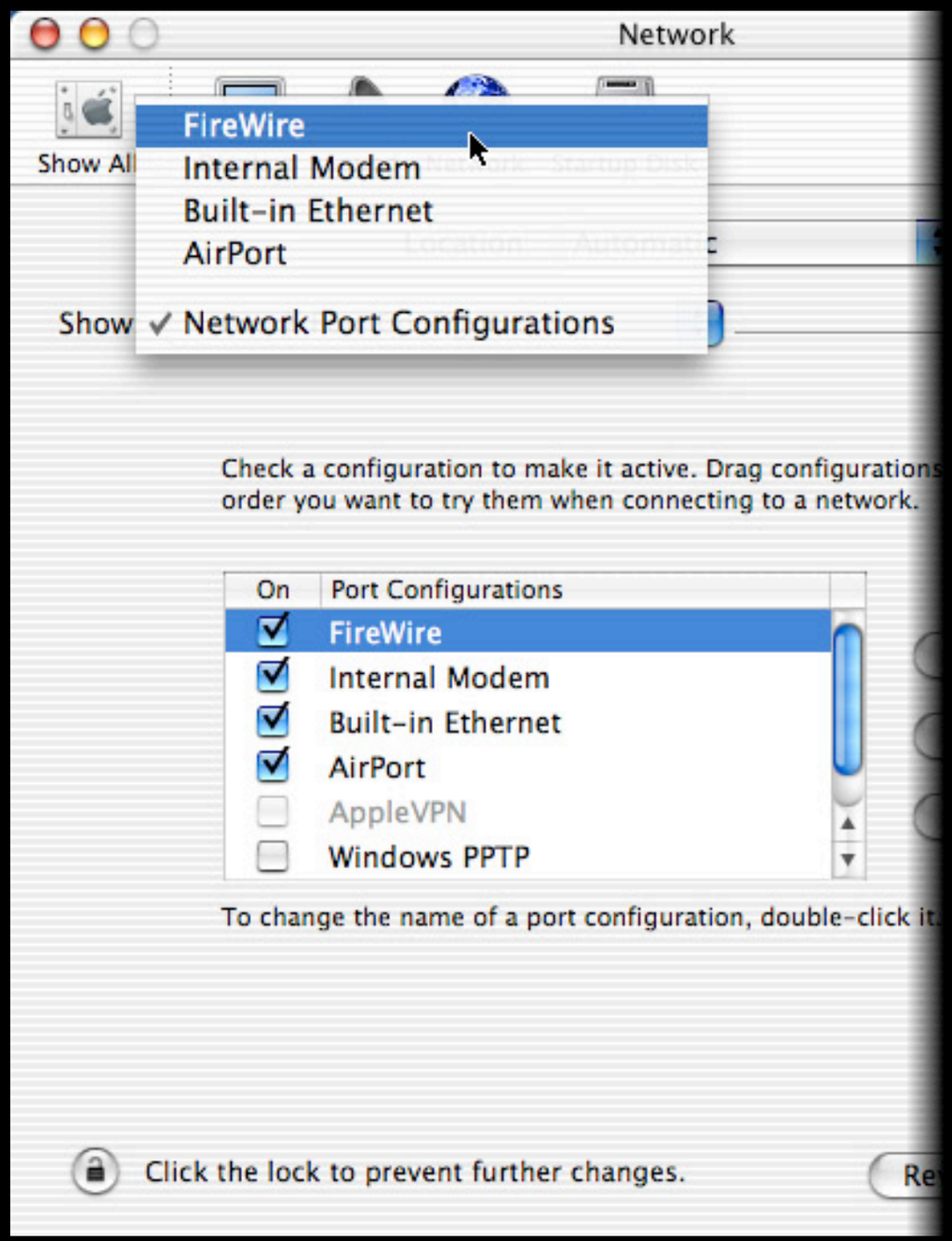
Internet Protocol (TCP/IP)

- IEEE RFC 2734 specifies how to transport IP on IEEE 1394
  - http://www.ieee.org
- Internet datagrams are sent in FireWire block writes
  - Each node allocates an address at which it receives IP
  - This address is published in the Configuration ROM
- Broadcast packets (e.g. ARP) are sent in FireWire asynch stream packets on channel 31
  - IEEE 1394a reserves channel 31 for this use (and others)

Using IP Over FireWire

- Any Internet service can operate over FireWire
  - HTTP, FTP, AFP, etc.
- Personal file sharing: No reboot required
  - Faster than TDM? May depend on usage pattern
- Cluster computing: IP is commonly used
  - But native FireWire memory access could be faster

Configuring IP



- Network Preferences panel
- Net. Port Configurations
- Turn on FireWire
  - Or Ethernet Adaptor (enX)
    - Rename if desired
- Drag to top of list to use Rendezvous Auto-IP
- Select FireWire port
- Configure as desired

Apple Support for IP Over FireWire

- IOFireWireIP.kext implements RFC 2734
  - Developer preview releases (ADC)
  - Mac OS X Server 10.2.5
  - Pre-installed on XServe
  - Standard in Mac OS X 10.3, with full user interface
- New in Mac OS X 10.3
  - IPv6 (per RFC 3146)
  - Improved performance

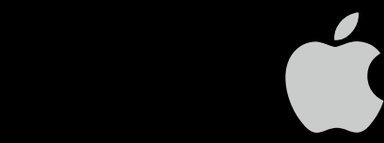
IP Performance on FireWire

- Ethernet hardware is designed for maximum IP speed
  - FireWire hardware is multi-purpose, not specialized for IP
- IP over FireWire on XServe, measured with NetPerf
  - Mac OS X 10.2.4 and FireWire 400: 216 mbps
  - Mac OS X 10.2.5 and FireWire 400: 277 mbps
- Preliminary results for same tests
  - Mac OS X 10.3 and FireWire 400: 317 mbps
  - Mac OS X 10.3 and FireWire 800: 660 mbps

Isochronous Services

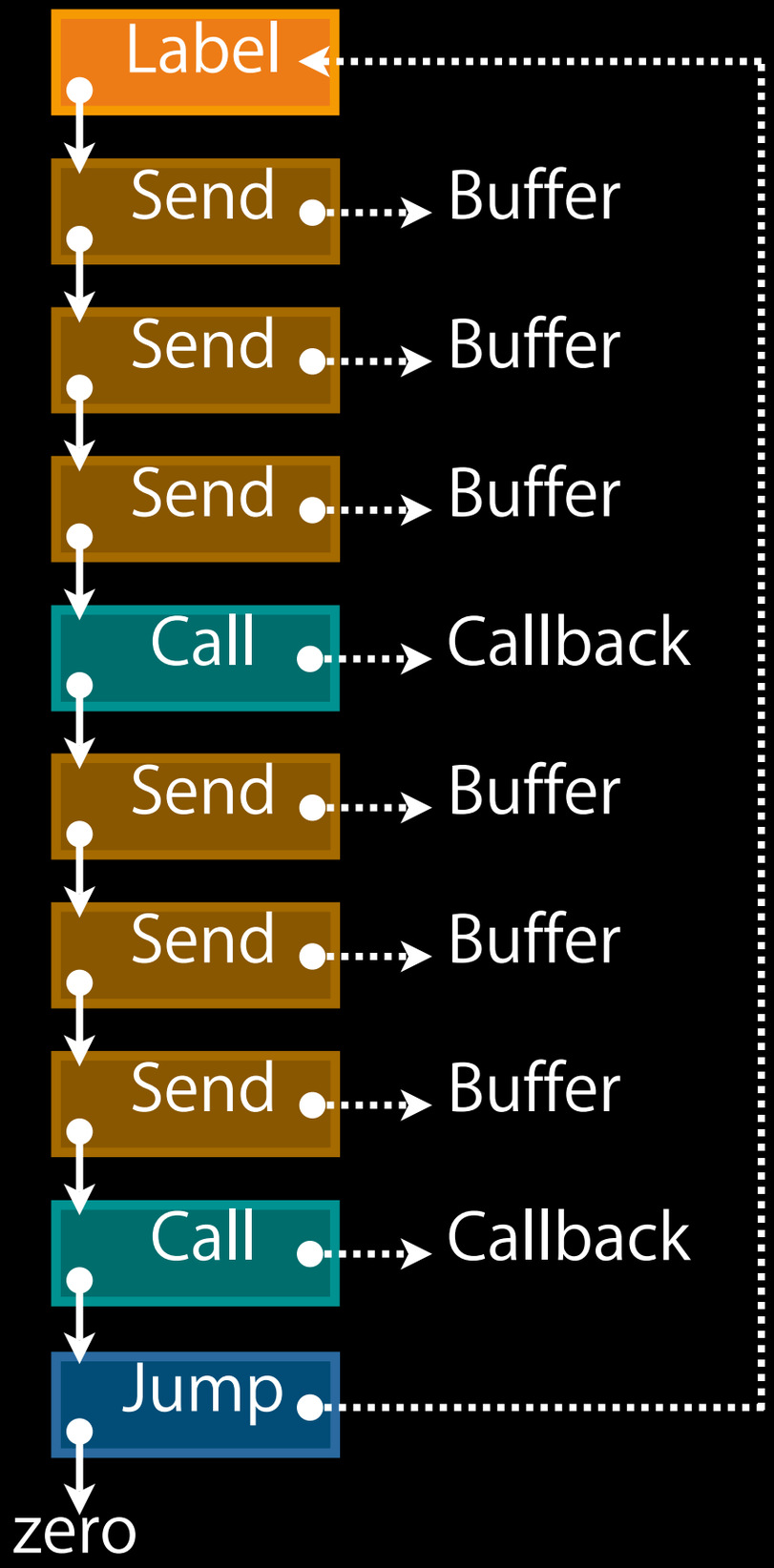
DCL—Datastream Control Language

- Isochronous transfer moves 8,000 packets/second
  - Moving packets by system calls would be inefficient
  - Instead, let DMA move packets at regular intervals
- Isochronous DMA is complicated
  - Register set and programming vary from Mac to Mac
  - Operation requires physical memory addresses
- DCLs: Abstract, packet-by-packet transfer instructions
  - Precise, yet efficient and hardware-independent



DCL Concepts

- A DCL program is a linked list of operations that will execute in order
- Each operation does one thing, such as:
  - Send or receive a packet (never both in one program)
  - Invoke a callback to advise software of DCL status
  - Jump to another DCL (out of linked-list order)
- A DCL program will be compiled to execute natively on whatever Link hardware is present
  - Changes to a running DCL program are restricted

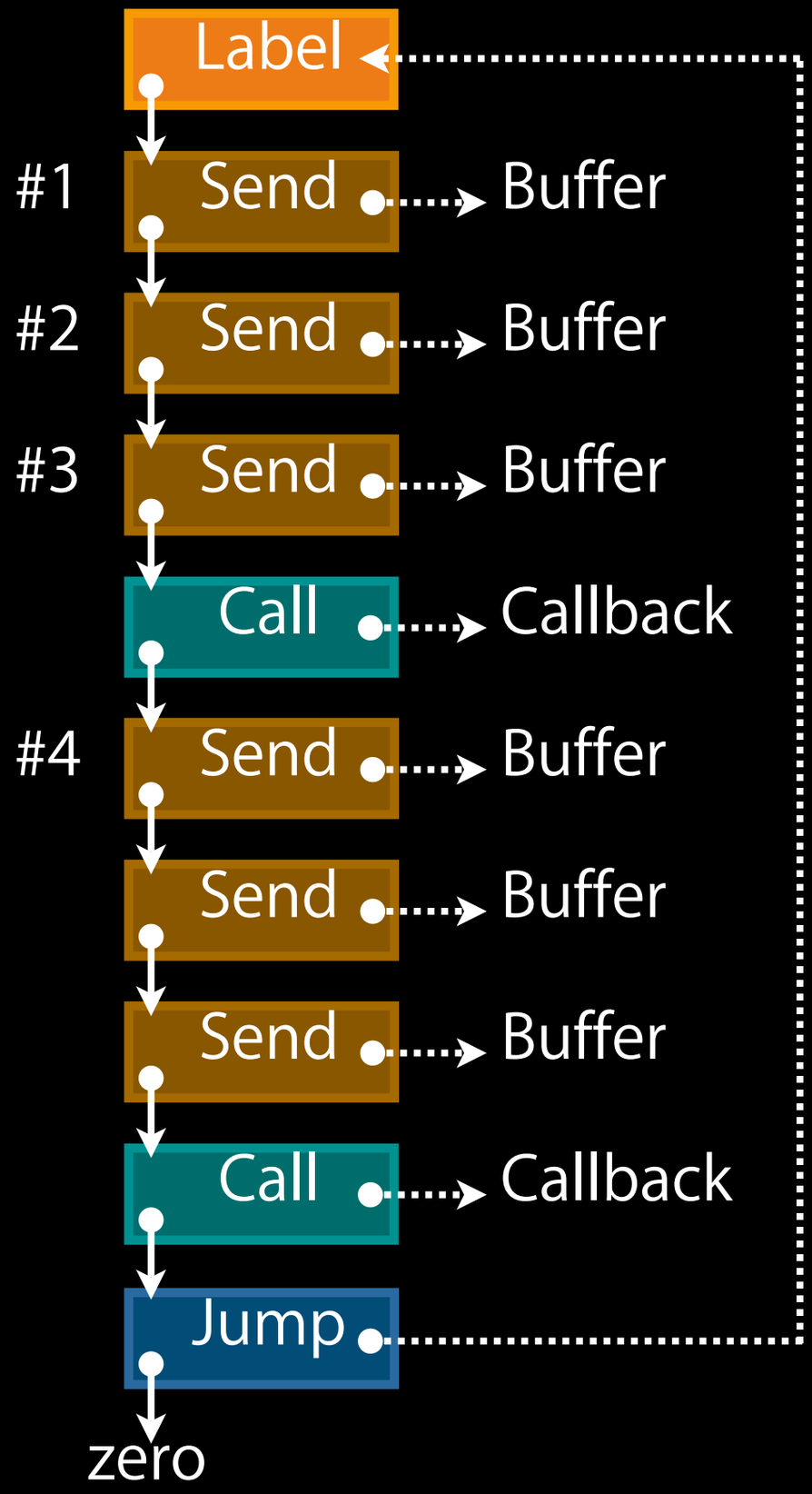


DCL Example

- This program sends data continuously (in a loop)
- Each callback tells the driver (not shown) to fill in new data for the next pass
- The DCL linked list always ends in zero, even for loops
  - Use Jump/Label to loop

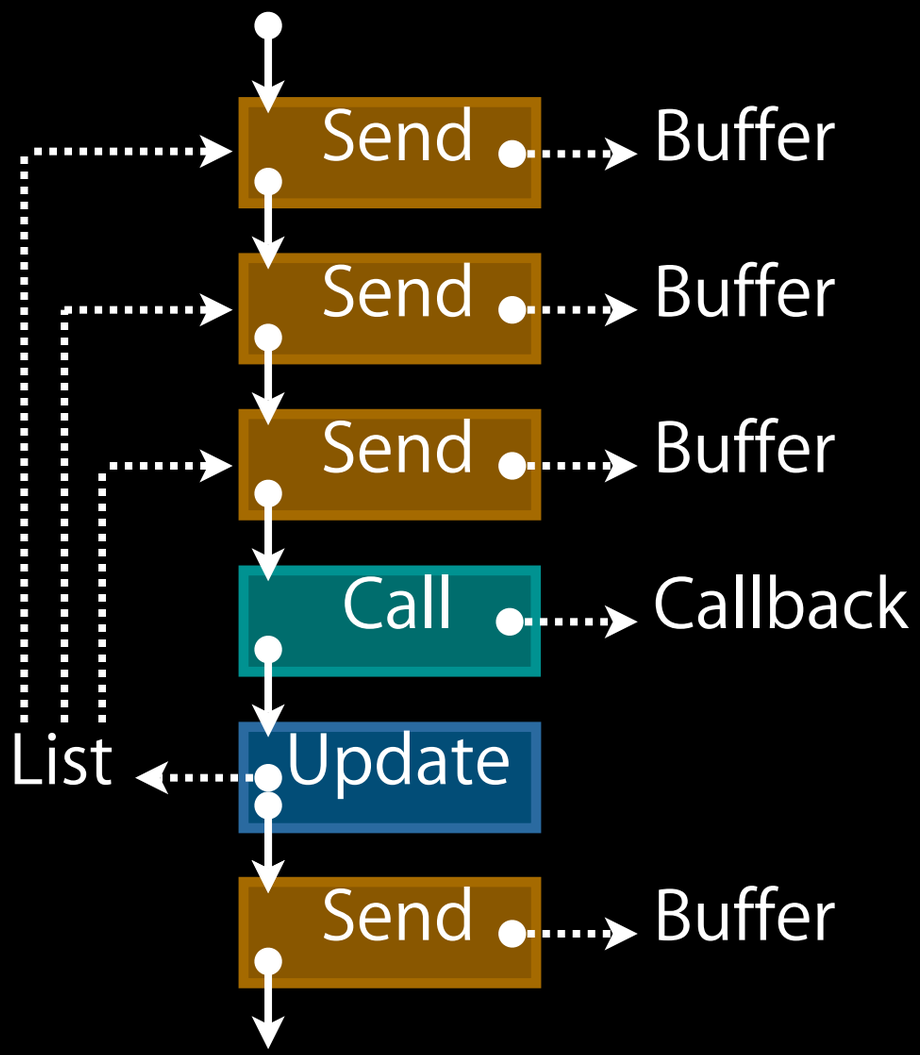
Linked-list “next” pointers

Other pointers



DCL Example

- DMA does not “wait” for callbacks, jumps, or labels
- Sends #1, #2, #3, and #4 all happen on consecutive cycles (~125µs apart)
- Callback must refresh buffers #1, #2, and #3 before the DMA loops back to Send #1



Update DCLs

- FireWire needs to know when a buffer has been changed
  - To byte-swap headers, etc.
- Update operation points to a list of DCLs that have changed
  - FireWire knows what kind of update is needed based on the opcodes of the DCLs pointed to by the list

Other DCL Techniques

- Non-looping program
  - Driver changes jumps on the fly so DMA never loops
  - If driver keeps up, DMA never reaches program end
  - Safer than a fixed loop; never sends data twice
  - See DV driver transmit DCL
- Timestamp operation
  - Records the 1394 cycle timer as the DCL runs
  - Must be updated before software inspects it

New-Style DCLs in Mac OS X 10.3

- 1394 Open HCI is now standard
- Old-style DCL programs had many rules reflecting OHCI
  - Label must be followed by send or receive
  - Jump cannot immediately follow Label
- New-style DCL reflects OHCI and has new features
  - One DCL per cycle
  - Each DCL includes optional Jump, Callback, Update, etc.
  - New options to get error status and change data length
- Old-style still supported, but deprecated

New AVC Services

AVC Controller Enhancements

- New plug APIs in IOFireWireAVCLibUnitInterface

```
IOReturn (*makeP2PInputConnection) (void * self,
   UInt32 inputPlug,
   UInt32 chan);

IOReturn (*breakP2PInputConnection) (void * self,
   UInt32 inputPlug);

IOReturn (*makeP2POutputConnection) (void * self,
   UInt32 outputPlug,
   UInt32 chan,
   IOFWSpeed speed);

IOReturn (*breakP2POutputConnection) (void * self,
   UInt32 outputPlug);
```

AVC Protocol Enhancements

- New APIs in IOFireWireAVCLibProtocol
- Publish AVC Unit Directory and add Subunits

```
IOReturn (*publishAVCUnitDirectory) (void * self);

IOReturn (*addSubunit) (void *self,
    UInt32 subunitType,
    UInt32 numSourcePlugs,
    UInt32 numDestPlugs,
    void *refCon,
    IOFWAVCSubunitPlugHandlerCallback callback,
    UInt32 *pSubunitTypeAndID);
```

Handling AVC Commands

- New APIs in IOFireWireAVCLibProtocol

```
IOReturn (*installAVCCommandHandler) (void *self,
   UInt32 subUnitTypeAndID,
   UInt32 opCode,
   void *refCon,
   IOFWAVCCommandHandlerCallback callback);

IOReturn (*sendAVCResponse) (void *self,
   UInt32 generation,
   UInt16 nodeID,
   const char *response,
   UInt32 responseLen);
```

AVC Unit Commands Handled

- All can be overridden (with care) by your own handler

PLUG INFO	0x02		Status
OUTPUT PLUG SIGNAL FORMAT	0x18	Control	Status
INPUT PLUG SIGNAL FORMAT	0x19	Control	Status
CONNECTIONS	0x22		Status
CONNECT	0x24	Control	Status
DISCONNECT	0x25	Control	
UNIT INFO	0x30		Status
SUBUNIT INFO	0x31		Status
POWER	0xB2	Control	Status

Existing AVC APIs

```
AddCallbackDispatcherToRunLoop (…)
removeCallbackDispatcherFromRunLoop (…)
setMessageCallback (…)
allocateInputPlug (…)
freeInputPlug (…)
readInputPlug (…)
updateInputPlug (…)                 //
allocateOutputPlug (…)              // All supported
freeOutputPlug (…)                  //
readOutputPlug (…)
updateOutputPlug (…)
readOutputMasterPlug (…)
updateOutputMasterPlug (…)
readInputMasterPlug (…)
updateInputMasterPlug (…)

setAVCRequestCallback (…)           // Deprecated
```

New SBP-3 Support

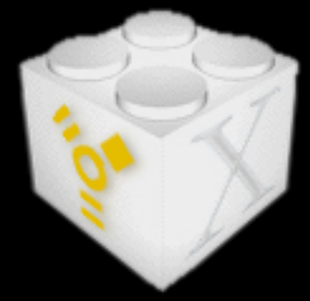
SBP-3 Fast Start

- SBP-2 requires up to five packets to start an I/O
  - ORB pointer, ORB fetch, ORB, page table fetch, page table
- Fast Start sends all this information in a single packet
  - Measured speedup is around 30% for small I/Os
- SBP-3 is in ballot within ANSI T-10
- Support available in Mac OS X 10.2.4 and later
  - Requires Configuration ROM addition
  - Requires support in drive firmware
  - Contact Apple if your hardware supports SBP-3

FireWire Resources

Software Development Kits

- FireWire SDK 16 for Mac OS X
  - Sample code
  - Tools, utilities, and test applications
  - Pre-release FireWire components (sometimes)
  - Instructions, notes, documentation, links
- Free public download



http://developer.apple.com/sdk

FireWire Reference Platform

- FireWire service and driver stack for embedded use
  - Fully independent of Mac OS X FireWire
- Suitable for single or multiple function devices
  - Camera, television, set-top box, printer, etc.
- Modular architecture supports common protocols
  - SBP-2 Target and Initiator; many AV/C command sets

http://developer.apple.com/firewire/platform

FireWire Developer Events

- Plugfests at Macworld and WWDC
- Developer Kitchens
  - 2-3 day workshop at Apple with FireWire engineers
  - Often held in conjunction with a new SDK
  - Located in Tokyo and Cupertino as needed
  - Contact Apple Developer Relations when ready
- 1394 Trade Association Interoperability Workshops

1394 TA Interop

- Formal testing three ways
  - Pairwise private tests
  - Official 1394 TA test suites
  - “Grand Melee”
- Four times each year
  - Tokyo, Taipei, USA x 2
- Open to non-members

http://www.1394ta.org

1394 Trade Association

- Industry group for promotion of IEEE 1394
- Hosts mailing lists, web site, and quarterly meetings for technical working groups and standards development
- Hosts interop workshops and developers conference
- Organizes PR activities and trade show exhibits
- Source of many FireWire specifications
  - AV/C commands sets, IIDC camera, Automotive, Testing

http://www.1394ta.org

FireWire Tools

What Exactly Is a PHY?

- PHY is short for PHYsical Layer
- The PHY drives different voltages on the FireWire cable to make serial bits
- The PHY acts as a repeater, and performs arbitration
- Every node has a PHY—even hubs
- In comparison, the Link operates at a higher level
  - Packets, CRCs, DMA, Cycle Timer
  - OHCI: Open Host Controller Interface (in all Macs today)

1394a PHY Register Map

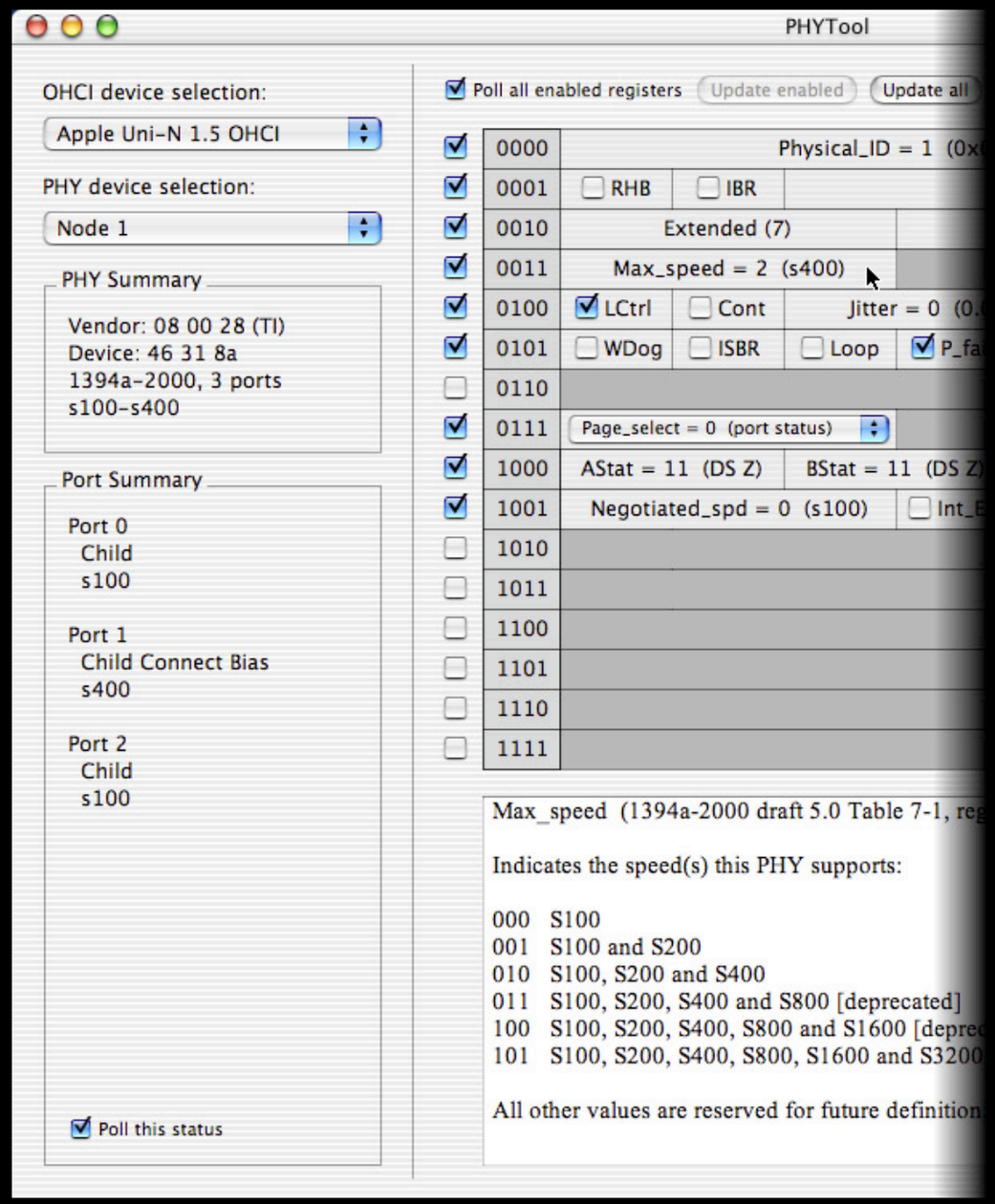
Address	0	1	2	3	4	5	6	7
0000_2	Physical_ID						R	PS
0001_2	RHB	IBB	Gap_count					
0010_2	Extended (7)			Total_ports				
0011_2	Max_speed				Delay			
0100_2	LCtrl	Contender	Jitter			Pwr_class		
0101_2	Watchdog	ISBR	Loop	Pwr_fail	Timeout	Port_event	Enab_accel	Enab_multi
0110_2								
0111_2	Page_select				Port_select			
1000_2	Register0 *Page_select*							
⋮	⋮							⋮
1111_2	Register7 *Page_select*							

Figure 7-1: Extended PHY Register Map for the Cable Environment

1394b PHY Register Map

Address	0	1	2	3	4	5	6	7
0000_2	Physical_ID						R	PS
0001_2	RHB	IBB	Gap_count					
0010_2	Extended (7)			Total_ports				
0011_2	Max_speed			Enable_ standby	Delay			
0100_2	LCtrl	Contender	Jitter			Pwr_class		
0101_2	Watchdog	ISBR	Loop	Pwr_fail	Timeout	Port_event	Enab_accel	Enab_multi
0110_2	Max_legacy_path_speed			B_Link	Bridge			
0111_2	Page_select				Port_select			
1000_2	Register0 *Page_select*							
⋮	⋮							⋮
1111_2	Register7 *Page_select*							

Figure 12-1: Extended PHY Register Map for the Cable Environment



PHYTool

- Real-time PHY register browser and editor
- Integrated documentation
- Remote PHY access
- PHY and port status summary
- Works on all 1394 OHCI interfaces

PHYTool

OHCI device selection:

Apple Uni-N 1.5 OHCI

PHY device selection:

Node 1

PHY Summary

Vendor: 08 00 28 (TI)
Device: 46 31 8a
1394a-2000, 3 ports
s100-s400

Port Summary

Port 0
Child
s100

Port 1
Child Connect Bias
s400

Port 2
Child
s100

Poll this status

Poll all enabled registers | Update enabled | Update all | Reset PHY-Link interface | Standard: 1394a-2000

☑	0000	Physical_ID = 1 (0x01)						Root = 1	PS = 1	=07	3f	Set
☑	0001	☐ RHB	☐ IBR	Gap_count = 63 (0x3f)						=3f	3f	Set
☑	0010	Extended (7)		Total_ports = 3						=e3	3f	Set
☑	0011	Max_speed = 2 (s400)				Delay = 0 (144 ns)				=40	3f	Set
☑	0100	☑ LCtrl	☐ Cont	Jitter = 0 (0.020 µs)		Pwr_class = 4 (100: ≤3W)				=84	3f	Set
☑	0101	☐ WDog	☐ ISBR	☐ Loop	☑ P_fail	☐ Tmout	☐ PEvent	☐ Accel	☐ Multi	=10	3f	Set
☐	0110									=??	3f	Set
☑	0111	Page_select = 0 (port status)				Port_select = 0				=00	3f	Set
☑	1000	AStat = 11 (DS Z)	BStat = 11 (DS Z)	Child = 1	Conn = 0	Bias = 0	☐ Disab			=f8	3f	Set
☑	1001	Negotiated_spd = 0 (s100)		☐ Int_En	☐ Fault					=00	3f	Set
☐	1010									=??	3f	Set
☐	1011									=??	3f	Set
☐	1100									=??	3f	Set
☐	1101									=??	3f	Set
☐	1110									=??	3f	Set
☐	1111									=??	3f	Set

Max_speed (1394a-2000 draft 5.0 Table 7-1, register 3, bits 0-2)

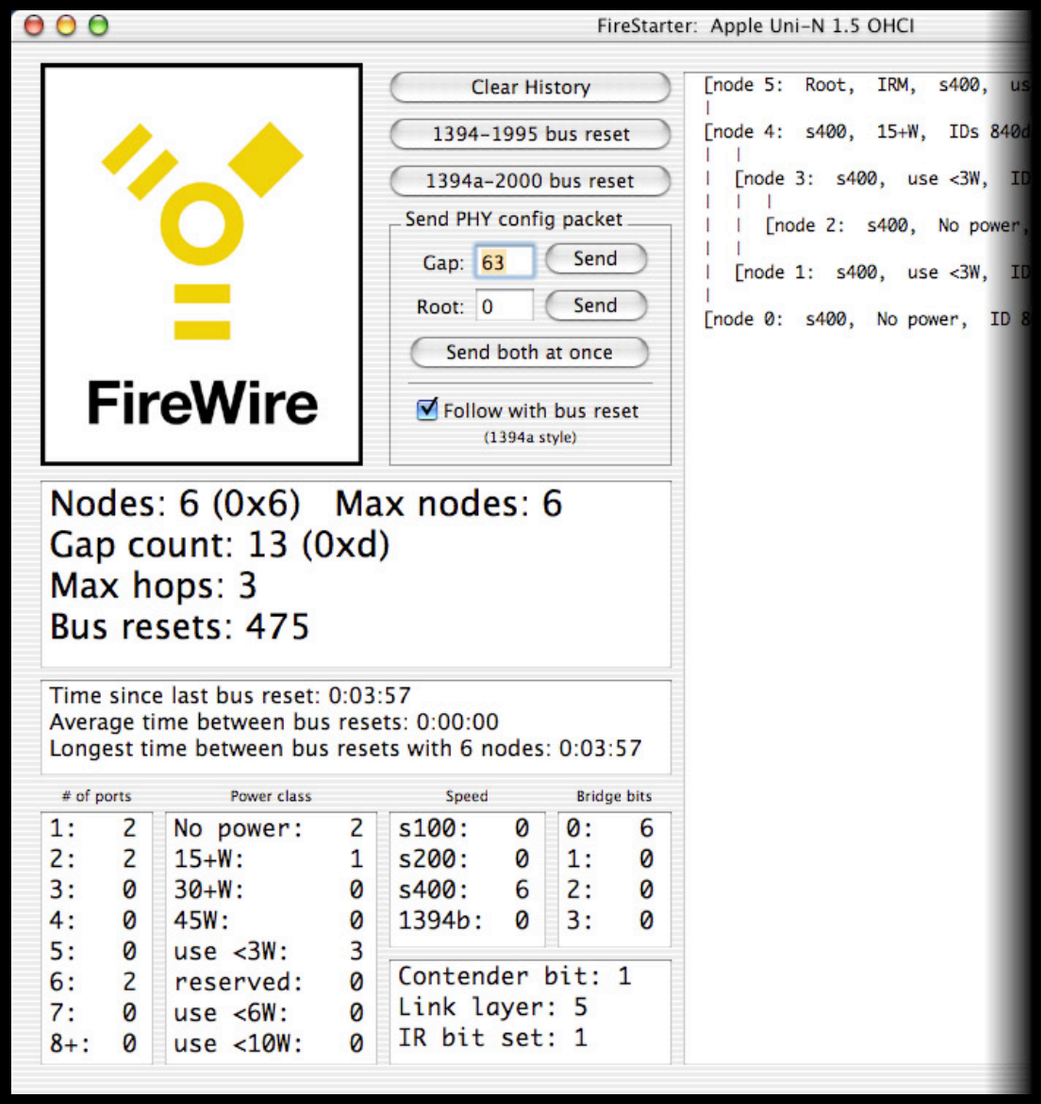
Indicates the speed(s) this PHY supports:

000 S100
001 S100 and S200
010 S100, S200 and S400
011 S100, S200, S400 and S800 [deprecated]
100 S100, S200, S400, S800 and S1600 [deprecated]
101 S100, S200, S400, S800, S1600 and S3200 [deprecated]

All other values are reserved for future definition.

FireWire

FireStarter



- Real-time topology display
  - Much faster than commercial analyzers
- Bus summary by node attribute
  - Speed, power, ports, etc.
- Bus configuration tools
  - Bus reset; Set root/gap
- Works on all 1394 OHCI interfaces

FireStarter: Apple Uni-N 1.5 OHCI

FireWire

Clear History

1394-1995 bus reset

1394a-2000 bus reset

Send PHY config packet

Gap: 63 Send

Root: 0 Send

Send both at once

Follow with bus reset

(1394a style)

```
[node 5:  Root,  IRM,  s400,  use <3W,  ID 854d8cd2]  *IR*
|
[node 4:  s400,  15+W,  IDs 840d816d 84817000]
|  |
|  [node 3:  s400,  use <3W,  IDs 834d849d 8381d000]
|  |  |
|  |  [node 2:  s400,  No power,  ID 824d8080]
|  |
|  [node 1:  s400,  use <3W,  ID 814d8460]
|
[node 0:  s400,  No power,  ID 804d8080]
```

Nodes: 6 (0x6) Max nodes: 6
Gap count: 13 (0xd)
Max hops: 3
Bus resets: 475

Time since last bus reset: 0:03:57
Average time between bus resets: 0:00:00
Longest time between bus resets with 6 nodes: 0:03:57

# of ports		Power class		Speed		Bridge bits	
1:	2	No power:	2	s100:	0	0:	6
2:	2	15+W:	1	s200:	0	1:	0
3:	0	30+W:	0	s400:	6	2:	0
4:	0	45W:	0	1394b:	0	3:	0
5:	0	use <3W:	3				
6:	2	reserved:	0				
7:	0	use <6W:	0				
8+:	0	use <10W:	0				

Contender bit: 1
Link layer: 5
IR bit set: 1

FireStarter: Apple Pangea OHCI

FireWire

Clear History

1394-1995 bus reset

1394a-2000 bus reset

Send PHY config packet

Gap: 63 Send

Root: 9 Send

Send both at once

Follow with bus reset

(1394a style)

Nodes: 63 (0x3f) Max nodes: 63
Gap count: 63 (0x3f)
Max hops: 11
Bus resets: 4650

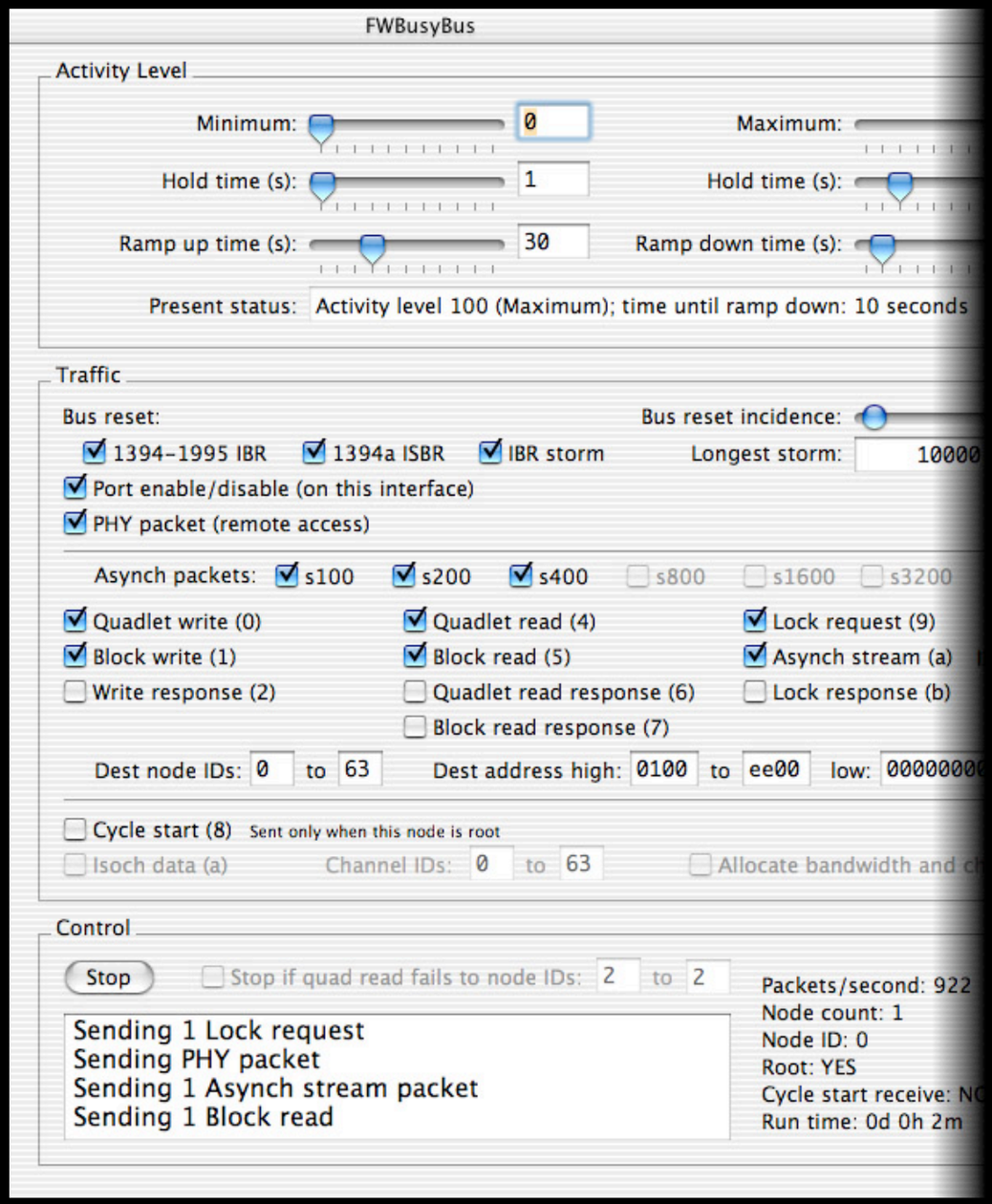
Time since last bus reset: 0:01:04
Average time between bus resets: 0:00:00
Longest time between bus resets with 63 nodes: 0:01:04

# of ports		Power class		Speed		Bridge bits	
1:	5	No power:	26	s100:	3	0:	63
2:	19	15+W:	2	s200:	1	1:	0
3:	33	30+W:	0	s400:	58	2:	0
4:	1	45W:	0	1394b:	1	3:	0
5:	0	use <3W:	26				
6:	5	reserved:	3				
7:	0	use <5W:	0				
8+:	0	use <9W:	6				

Contender bit: 14
Link layer: 42
IR bit set: 1

```
[node 62 (3e): Root, IRM, s200, No power, ID be7f48d4]
[node 61 (3d): s400, 15+W, ID bd7f81ec]
|  [node 60 (3c): s400, use <3W, IDs bc7f84f9 bc83f000]
|  |  |  |  |  [node 59 (3b): s400, use <9W, ID bb3f876c]
|  |  |  |  |  [node 58 (3a): s400, use <9W, ID ba3f8764]
|  |  |  |  [node 57 (39): s400, use <3W, ID b97f8460]
|  |  |  [node 56 (38): s400, use <3W, ID b83f84ec]
|  |  |  |  [node 55 (37): s400, No power, ID b77f8094]
|  |  |  [node 54 (36): s400, No power, ID b67f8890]
|  |  [node 53 (35): s400, No power, ID b57f80b4]
|  |  [node 52 (34): s400, No power, ID b43f80e4]
|  |  [node 51 (33): s100, No power, ID b37f0880]
|  [node 50 (32): s400, 15+W, IDs b23f81ed b283f000]
|  |  |  |  |  [node 49 (31): s400, use <3W, IDs b13f84bd b183f000]
|  |  |  |  |  |  |  |  |  [node 48 (30): s400, No power, ID b07f8080]
|  |  |  |  |  |  |  |  [node 47 (2f): s400, use <3W, ID af3f8494]
|  |  |  |  |  |  |  [node 46 (2e): s400, RESERVED, ID ae3f85d8]
|  |  |  |  |  |  |  [node 45 (2d): s400, use <3W, ID ad3f849c]
|  |  |  |  |  |  |  [node 44 (2c): s400, No power, ID ac7f8094]
|  |  |  |  |  |  [node 43 (2b): s400, use <3W, ID ab3f84bc]
|  |  |  |  |  |  |  [node 42 (2a): s400, No power, ID aa7f8094]
|  |  |  |  |  |  [node 41 (29): s400, use <3W, ID a97f84e0]
|  |  |  |  |  |  [node 40 (28): s400, use <3W, ID a83f8cbc]
|  |  |  |  |  |  |  [node 39 (27): s400, use <3W, ID a77f8490]
|  |  |  |  |  |  [node 38 (26): s400, use <9W, ID a67f87b0]
|  |  |  |  |  |  [node 37 (25): s400, use <3W, ID a53f849c]
|  |  |  |  |  |  [node 36 (24): s400, No power, ID a47f8094]
|  |  |  |  |  [node 35 (23): s400, No power, ID a33f80bc]
|  |  |  |  |  |  [node 34 (22): s100, No power, ID a27f0880]
|  |  |  |  |  [node 33 (21): s400, No power, ID a17f80e4]
|  |  |  |  |  [node 32 (20): s400, No power, ID a07f8094]
|  |  |  |  [node 31 (1f): s400, use <3W, IDs 9f3f84fd 9f83e000]
|  |  |  |  |  |  |  |  [node 30 (1e): s400, use <3W, ID 9e3f84b0]
|  |  |  |  |  |  |  |  [node 29 (1d): s400, use <3W, ID 9d3f8c64]
|  |  |  |  |  |  |  [node 28 (1c): s400, use <3W, ID 9c7f8ce0]
|  |  |  |  |  |  |  [node 27 (1b): s400, No power, ID 9b7f8894]
|  |  |  |  |  |  [node 26 (1a): s400, use <3W, ID 9a7f84e0]
|  |  |  |  |  |  [node 25 (19): s400, use <3W, ID 997f84b0]
|  |  |  |  |  |  [node 24 (18): s100, No power, ID 987f0880]
|  |  |  |  |  [node 23 (17): s400, RESERVED, ID 973f85d8]
|  |  |  |  |  [node 22 (16): s400, use <3W, ID 967f8480]
|  |  |  |  [node 21 (15): s400, No power, ID 957f88b4]
|  |  |  |  [node 20 (14): s400, use <9W, ID 947f8760]
|  |  |  [node 19 (13): s400, use <3W, ID 937f84ec]
|  |  |  |  [node 18 (12): s400, No power, ID 923f80ec]
|  |  |  |  |  [node 17 (11): 1394b, use <3W, ID 917fc464]
|  |  |  |  [node 16 (10): s400, No power, ID 907f8094]
|  |  |  [node 15 (f): s400, No power, ID 8f7f8894]
|  |  [node 14 (e): s400, use <3W, IDs 8e3f846d 8e83f000]
|  |  |  |  [node 13 (d): s400, No power, ID 8d7f8892]
|  |  |  [node 12 (c): s400, use <3W, ID 8c7f84e0]
|  |  |  [node 11 (b): s400, No power, ID 8b7f8090]
|  |  [node 10 (a): s400, use <3W, ID 8a7f8478]
|  |  [node 9: s400, use <3W, ID 897f8490]
|  [node 8: s400, use <3W, ID 883f84ec]
|  |  [node 7: s400, No power, ID 877f8094]
```

FWBusyBus



- Traffic generator
  - Asynchronous packets
  - Bus resets and storms
  - Port enables and disables
- Automatic rate adjuster
- Speed selection
- Works on all 1394 OHCI interfaces

FWBusyBus

OHCI device selection:

Apple Uni-N 1.5 OHCI

PHY Summary

Vendor: 08 00 28 (TI)
Device: 42 42 96
1394a-2000, 1 port
s100-s400

Port Summary

Port 0
Child DIS
s100

Activity Level

Minimum: 0 Maximum: 100
Hold time (s): 1 Hold time (s): 20
Ramp up time (s): 30 Ramp down time (s): 10
Present status: Activity level 100 (Maximum); time until ramp down: 10 seconds

Traffic

Bus reset: Bus reset incidence: 5
1394-1995 IBR 1394a ISBR IBR storm Longest storm: 10000 microseconds
Port enable/disable (on this interface)
PHY packet (remote access)

Asynch packets: s100 s200 s400 s800 s1600 s3200

Quadlet write (0) Quadlet read (4) Lock request (9)
Block write (1) Block read (5) Asynch stream (a) IDs: 11 to 29
Write response (2) Quadlet read response (6) Lock response (b)
Block read response (7)
Channels will NOT be allocated

Dest node IDs: 0 to 63 Dest address high: 0100 to ee00 low: 00000000 to fffffffc

Cycle start (8) Sent only when this node is root
Isoch data (a) Channel IDs: 0 to 63 Allocate bandwidth and channel

Control

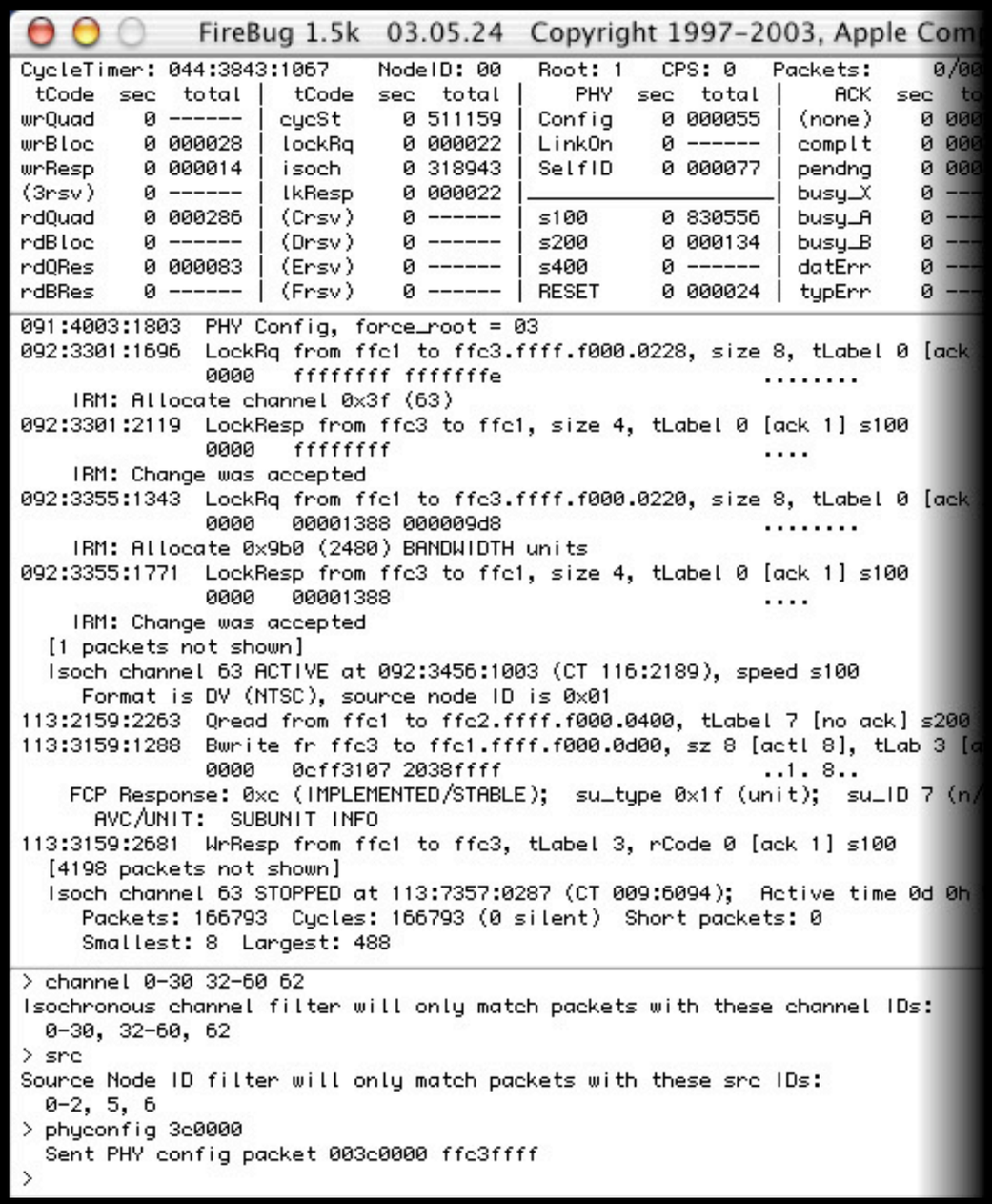
Stop Stop if quad read fails to node IDs: 2 to 2

Sending 1 Lock request
Sending PHY packet
Sending 1 Asynch stream packet
Sending 1 Block read

Packets/second: 922
Node count: 1
Node ID: 0
Root: YES
Cycle start receive: NO
Run time: 0d 0h 2m

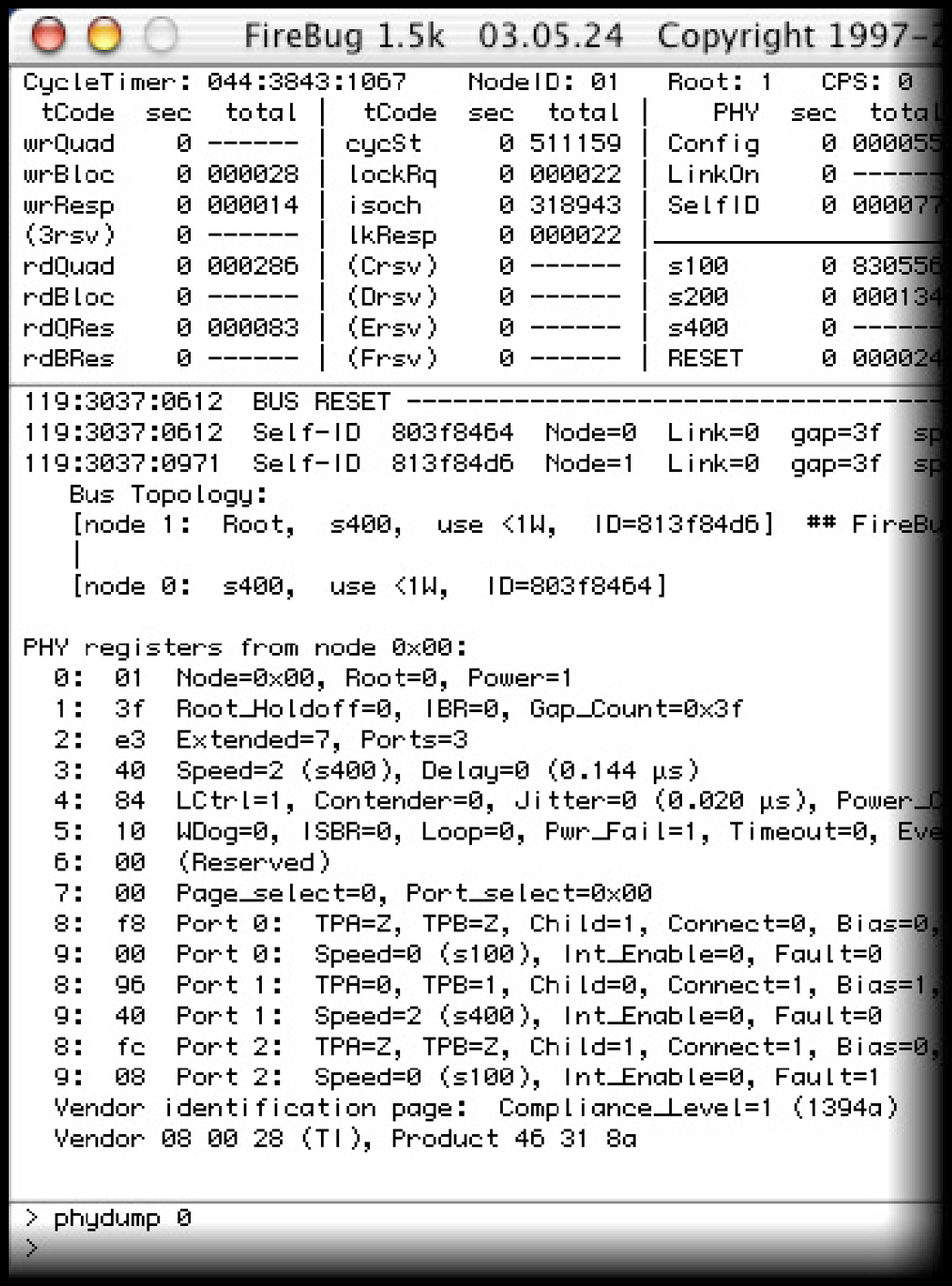
FireWire

FireBug



- Real-time packet snooper
  - Decodes all 1394 packets
  - Precision timestamps
  - Decodes SBP-2, AVC, IRM
  - Bus configuration services
- Requires TI PCI-Lynx interface
  - PCI or CardBus card
  - Built-in on some old Macs

FireBug Improved



- Better decode
  - More AVC commands
  - SBP-2 survives bus reset
  - PHY register dump
- New features include:
  - Node-ID filters
  - Isochronous summary
  - Dual-processor support

Demo—FireBug

http://developer.apple.com/wwdc2003/urls.html

New Mac OS X FireWire Tools

- PHYTool (any OHCI)
- FireStarter (any OHCI)
- FWBusyBus (any OHCI)
- FireBug (still needs PCI-Lynx)

- All tools run on Mac OS X 10.2.5 and later
- All tools displace other FireWire drivers and services
- Be careful—make backups before using tools

Available Today

FireWire SDK 17 for Mac OS X

http://developer.apple.com/sdk

this slide might be deleted

Wrap Up

Roadmap

504 Developing with FireWire 800 Writing applications and drivers for FireWire	**Pacific Heights** **Wed., 9:00am**
FF006 FireWire and USB One-stop shopping for all your Serial Bus needs	**North Beach** **Wed., 2:30pm**
508 FireWire Reference Platform FireWire software for embedded systems	**Marina** **Thur., 3:30pm**
Plug and Unplug Fest FireWire, USB, and Bluetooth	**Apple Campus** **Thur., 7:30pm**

Who to Contact

Prashant Kanhere
Program Manager, FireWire Technologies
prashant@apple.com

Guillermo Ortiz
Apple Developer Relations
gao@apple.com

FireWire developer mailing list
www.lists.apple.com/mailman/listinfo/firewire
firewire@lists.apple.com

FireWire Reference Platform developer mailing list
www.lists.apple.com/mailman/listinfo/firewire-reference-platform
firewire-reference-platform@lists.apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Working with FireWire Device Interfaces
http://developer.apple.com/techpubs/macosx/Darwin/IOKit/iokit.html

Software Development Kit

FireWire SDK 17 for Mac OS X

http://developer.apple.com/hardware/FireWire

Standards and Specifications

Institute of Electrical and Electronics Engineers
http://www.ieee.org

1394 Trade Association
http://www.1394ta.org

International Standards Organization (ISO/IEC)

http://www.iec.ch

Q&A

Prashant Kanhere
Program Manager—FireWire Technologies
prashant@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference